(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 March 2001 (29.03.2001)

PCT

(10) International Publication Number
WO 01/22683 A2

(51) International Patent Classification[7]: H04L 29/06

(21) International Application Number: PCT/GB00/03684

(22) International Filing Date:
25 September 2000 (25.09.2000)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
99307551.4 24 September 1999 (24.09.1999) EP

(71) Applicant *(for all designated States except US)*: **BRITISH TELECOMMUNICATIONS PUBLIC LIMITED COMPANY** [GB/GB]; 81 Newgate Street, London EC1A 7AJ (GB).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **HOVELL, Peter** [GB/GB]; 24 Mill Road, Newbourne, Woodbridge, Suffolk, IP12 4NP (GB). **KING, John, Robert** [GB/GB]; 2 Hertfords Place, Chillesford, Woodbridge, Suffolk IP1 3SD (GB). **PATTERSON, John** [GB/GB]; The Annexe, 5 The Mills, Playford Road, Rushmere St Andrew, Ipswich, Suffolk, IP4 5RL (GB).

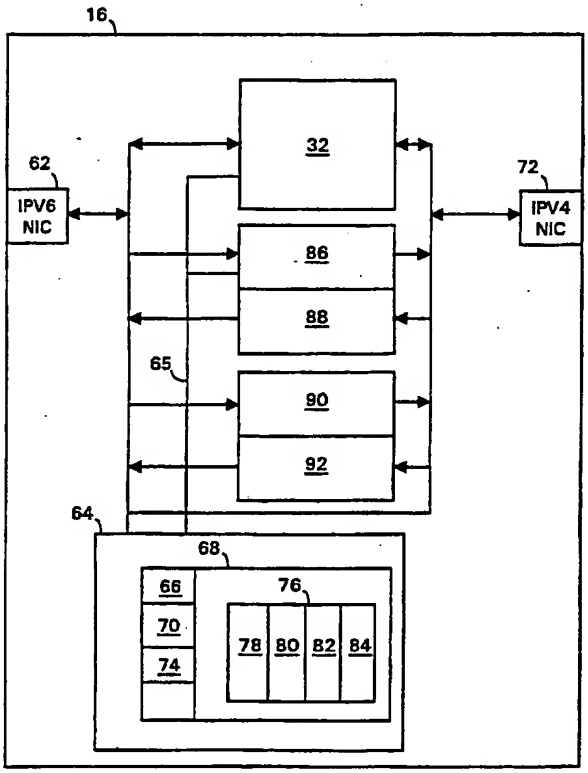
(74) Agent: **SEMOS, Robert, Ernest, Vickers**; BT Group Legal Services, Intellectual Property Department, Holborn Centre, 8th Floor, 120 Holborn, London EC1N 2TE (GB).

(81) Designated States *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CR, CU, CZ, DE, DK, DM, DZ, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZW.

(84) Designated States *(regional)*: ARIPO patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZW), Eurasian patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE), OAPI patent (BF, BJ, CF, CG, CI, CM, GA, GN, GW, ML, MR, NE, SN, TD, TG).

[Continued on next page]

(54) Title: PACKET NETWORK INTERFACING



(57) Abstract: A method of interfacing, and an interface for use, between two IPv6 domains separated by an IPv4 domain. The interface comprises a protocol converter, an encapsulator/un-encapsulator and a controller. When an IPv6 source wishes to send to a named destination in the other IPv6 domain, the source sends a normal IPv6 address request to its local DNS server, which relays it to an IPv6 name server in the other IPv6 domain. The response message, containing the true IPv6 address of the destination is received at the remote interface, which appends to the resulting protocol converted DNS response message a first additional record containing the true IPv6 address, and a second additional record containing the IPv4 address of that interface. Upon receipt at the other interface, the additional records are stripped off, their contents stored in an entry of a table, and the true IPv6 address written into the address record of the resulting IPv6 DNS response message. When the interface receives a packet from an IPv6 host, it checks whether the destination address matches an entry of its table, and if so sends the packet to the encapsulator together with the IPv4 address of the remote interface. The remote interface extracts the source address and the address of the encapsulating interface and stores these in an entry in its corresponding table for use in encapsulating return packets to the source. If, however, the destination address is recognised as being of IPv4-compatible or IPv4-mapped format, the packet is sent to a protocol converter.

WO 01/22683 A2

Published:

— *Without international search report and to be republished upon receipt of that report.*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

PACKET NETWORK INTERFACING

This invention relates to an interface between a first network operating in accordance with a first transmission protocol and having network addresses in accordance with a first addressing convention, herein referred to as first type addresses, and a second network operating in accordance with a second transmission protocol and having network addresses in accordance with a second addressing convention, herein referred to as second type addresses; and to the tunnelling of packets across the second network from one such interface to another such interface; and relates particularly, but not exclusively, to communication between hosts in respective Internet Protocol version 6 (IPv6) domains separated by an Internet Protocol version 4 (IPv4) domain.

Herein, the terms packet and message are used interchangeably and have the same meaning, and the term Internet domain is used as a specific example of a network.

In Internet technology, it had become apparent that the initial transport protocol, IPv4, needed to be enhanced, principally to increase the available address space and add a hierarchical address structure. The result was IPv6 which has a simplified header format compared with IPv4, but which uses 128 bit addresses as compared with 32 bit addresses used in IPv4.

Readers wishing to have an overview of this general transitional area might like to access a list of Internet-Drafts, which are working documents of the Internet Engineering Task Force (IETF), at http://www.ietf.org/1id-abstracts.txt, and a particularly relevant document is "A Guide to the Introduction of IPv6 in the IPv4 World" <draft-ietf-ngtrans-introduction-to-ipv6-transition-01.txt>, also referred to as "Guide to IPv6 Transition".

As mentioned, the present invention relates to tunnelling. Known tunnelling techniques are of two types: configured and automatic.

A configured tunnel is created by manual configuration of a tunnelling interface between an IPv6 domain and an IPv4 domain such that all packets received from the IPv6 domain are encapsulated in IPv4 packets addressed to a specific tunnel end point, i.e. the tunnelling interface between the IPv4 domain and the remote IPv6 domain containing the destination IPv6 host.

An automatic tunnel on the other hand does not need manual configuration: the tunnel end points are automatically determined. Several automatic tunnelling mechanisms are currently in development within the IETF, these being known in the art as, 6over4, 6to4, Dynamic Tunnelling, and Tunnel Broker.

For more detailed information on 6over4 the reader can obtain from the IETF, the document known as RFC2829, or any variant thereof, "Transmission of IPv6 over IPv4 Domains without Explicit Tunnels", by B. Carpenter and C. Jung, March 1999.

For more detailed information on 6to4 the reader can obtain from the IETF, the document known as draft-ietf-ngtrans-6to4-02.txt, or any variant thereof, "Connection of IPv6 Domains via IPv4 Clouds without Explicit Tunnels", by B. Carpenter and K. Moore.

For more detailed information on Dynamic Tunnelling the reader can obtain from the IETF, the document known as draft-ietf-ngtrans-dti-00.txt.

For more detailed information on Tunnel Broker the reader can obtain from the IETF, the document known as draft-ietf-ngtrans-broker-00.txt.

These known automatic tunnelling mechanisms use a variety of techniques to enable the tunnel to be automatically established:

- 6over4 Multicast
- 6to4 Special IPv6 address in which the top level aggregator (TLA) contains an identifier for the 6to4 tunnelling mechanism, and the next level aggregator (NLA) contains the IPv4 address of the tunnel end point
- Dynamic Tunnelling via DNS
- Tunnel Broker www based tool

According to a first aspect of the present invention, there is provided, an interface for use between a first network operating in accordance with a first transmission protocol and having network addresses in accordance with a first addressing convention, herein referred to as first type addresses, and a second network operating in accordance with a second transmission protocol and having network addresses in accordance with a second addressing convention, herein referred to as second type addresses, the interface having both a first type address and a second type address and comprising:

a protocol converter arranged to convert a message having a format in accordance with the first transmission protocol, herein referred to as a first type

message, into a message having a format in accordance with the second transmission protocol, herein referred to as a second type message, and vice versa;

means for encapsulating arranged to respond to receipt of a second type address together with a first type message by encapsulating that received first type message as the payload of a resulting encapsulating second type message, using that received second type address as the destination address of the resulting encapsulating second type message and using the second type address of the interface as the source address of the resulting encapsulating second type message;

means for un-encapsulating a second type message to retrieve its payload; and

an interface controller arranged to respond to receipt by the interface of a first type message from the first network by

examining the destination address of that first type message received from the first network,

sending to the protocol converter that first type message received from the first network if its destination address is of a first predetermined format,

else, deriving, directly or indirectly, from the destination address of that first type message received from the first network, a second type address for use by the encapsulating means as the destination address of a resulting encapsulating second type message, and

sending to the encapsulating means the derived second type address together with that first type message received from the first network.

Preferably, the controller is arranged to derive the second type address directly by retrieving it from a predetermined subaddress field of the destination address.

Alternatively, the controller is arranged to derive the second type address indirectly by accessing, in accordance with the destination address, a look-up table having entries in the form of a first type address associated with a second type address, and retrieving the second type address of an entry having a first type address matching the destination address.

Preferably, each entry of the address conversion table comprises a field for containing an identifier for identifying whether the controller is to send that first type

message received from the first network to the protocol converter or to the encapsulating means.

The encapsulating means may comprise a plurality of different encapsulators, each encapsulator being arranged to operate in accordance with a respective encapsulation type, and the controller being arranged to determine whether the destination address of that first type message received from the first network is of one of a respective corresponding plurality of predetermined formats, and, if so, to send that first type message received from the first network to the encapsulating means corresponding to that one predetermined format.

Preferably, each entry of the address conversion table comprises a field for containing an identifier for identifying a type of encapsulation whether the controller is to send that first type message received from the first network to the protocol converter or to the encapsulating means.

According to a second aspect of the present invention, there is provided, a method of operating an interface between a first network operating in accordance with a first transmission protocol and having network addresses in accordance with a first addressing convention, herein referred to as first type addresses, and a second network operating in accordance with a second transmission protocol and having network addresses in accordance with a second addressing convention, herein referred to as second type addresses, the method comprising the steps of:

examining the destination address of a first type message received from the first network; and

if the destination address of that received first type message is of a first predetermined format, protocol converting that received first type message;

else, encapsulating that received first type message in accordance with the second transmission protocol, using, as the destination address of a resulting encapsulating second type message, a second type address derived, directly or indirectly, from the destination address of that received first type message.

According to a third aspect of the present invention, there is provided, a method of operating an interface between a first network operating in accordance with a first transmission protocol and having network addresses in accordance with a first addressing convention, herein referred to as first type addresses, and a second network operating in accordance with a second transmission protocol and having

network addresses in accordance with a second addressing convention, herein referred to as second type addresses, the method comprising the steps of:

examining the destination address of a first type message received from the first network; and

if the destination address of that received first type message is of a first predetermined format, protocol converting that received first type message; and

if the destination address of that received first type message is of a second predetermined format, encapsulating that received first type message in accordance with the second transmission protocol, using, as the destination address of a resulting encapsulating second type message, a second type address derived, directly or indirectly, from the destination address of that received first type message.

Preferably, the second predetermined address format includes an identifier identifying an encapsulation type.

The first predetermined format may include a first predetermined portion whose content identifies that received first type message as suitable for protocol conversion.

The first predetermined format may also include a second predetermined portion whose content constitutes the second type address used as the destination address of a resulting encapsulating second type message.

Preferably, the second type address is derived directly by retrieving it from a predetermined subaddress field of the destination address.

Preferably, the examining step comprises the substeps of retrieving the destination address from the received first type message, and accessing a look-up table in accordance with the retrieved destination address.

More preferably, in methods for use when the look-up table comprises entries in the form of a first type address associated with a second type address, retrieval of the second type address of an entry having its first type address matching the destination address constitutes indirectly deriving the second type address from the destination address of that received first type message.

for use when the look-up table entries include a first identifier field containing an identifier identifying whether that first type message is to be protocol converted or encapsulated, there may be included the steps of retrieving the identifier from the first identifier field of the entry having its first type address matching the destination

address, and checking that the retrieved identifier is consistent with whichever of protocol conversion or encapsulation is to be performed upon that received first type message.

In methods for use when the look-up table entries include a second identifier field containing an identifier identifying an encapsulation type, and when there is a plurality of encapsulation types available, there may be included the steps of retrieving the identifier from the second identifier field of the entry having its first type address matching the destination address, and checking that the retrieved identifier is consistent with the type of encapsulation to be performed upon that received first type message.

Protocol converters are known for enabling an IPv6 host to send messages to an IPv4 host. When a new IPv6 host is activated in an IPv6 domain, it employs the technique known as Neighbourhood Discovery (ND) to find out the identity of hosts with whom it can communicate directly. It broadcasts an ND message containing its IPv6 network address, and each host that receives it sends a reply message containing that host's IPv6 network address. Since the domain uses an underlying transport mechanism, say Ethernet using media access control (MAC) addresses, each host receiving the ND message will retrieve the new host's IPv6 network address and also the MAC address of the new host, and the new host will retrieve from each reply message the sending host's IPv6 network address and its MAC address.

The new host now constructs an ND table in which each entry corresponds with a neighbouring host and comprises a first part in the form of the 128 bit IPv6 address of that neighbouring host, and a second part in the form of the associated MAC address.

The interface device (containing the protocol converter) between that IPv6 domain and an adjacent IPv4 domain will also have received the ND message and sent a reply message, and the new host will have made a special Default entry in the ND table having its first part formed by 128 zeros (in variants, these are all ones) and its second part formed by the MAC address of that interface device.

Thus, when that new host wants to send a message to one of the other hosts in its domain, it constructs an IPv6 message and accesses its ND table to retrieve the MAC address associated with the destination address. The message is

then encapsulated within an Ethernet packet in known manner and sent via the underlying Ethernet transport mechanism to the destination host.

If, on the other hand, the host constructs an IPv6 message having its destination address in the form of an IPv4-compatible or IPv4-mapped address, i.e. a message intended for an IPv4 host in the adjacent IPv4 domain, this destination address will not be found in the ND table. In this situation, the accessing algorithm will return the MAC address of the Default entry, and the message will be sent to the protocol converter.

Protocol converters can only convert (or translate) between corresponding fields of the headers of the IPv6 and IPv4 messages. Where a field in the header of, say, an IPv6 message has no corresponding field in the header of an IPv4 message, or vice versa, the information in that field will be lost in the protocol conversion process.

As mentioned above, tunnelling techniques are known for enabling IPv6 hosts to communicate amongst themselves when they are in spaced-apart domains. In this case, the interface device contains a tunnelling mechanism instead of a protocol converter. It will be appreciated that before now, for an IPv6 host to be able to communicate both with IPv4 hosts and with remote IPv6 hosts, it was necessary for that IPv6 host and the domain interface to be dual stack, i.e. having both IPv4 and IPv6 communication capability. If an IPv6 host was not dual stack, its accessing algorithm would return only a single MAC address for the Default entry This would be the MAC address of the input port of a protocol converter, if the network administration had decided that the IPv6 host is to be able to communicate with IPv4 hosts, or it would be the MAC address of the input port of a tunnelling mechanism, if the network administration had decided that the IPv6 host is to be able to communicate with IPv6 hosts. That Default entry MAC address could not be a common input address for both the protocol converter and the tunnelling mechanism.

Specific embodiments of the present invention will now be described with reference to the drawings in which:

Figure 1 is a schematic diagram of an IPv4 domain interfaced with two isolated IPv6 domains;

Figure 2 is a schematic diagram of a border router;

Figure 3 is a schematic diagram of an IPv6 DNS Response message;

Figure 4 is a schematic diagram of an IPv4 DNS Response message resulting from conversion of the IPv6 DNS Response message of Figure 3;

Figure 5 is a schematic diagram of an IPv6 DNS Response message resulting from conversion of the IPv4 DNS Response message of Figure 4; and

Figure 6 is a schematic diagram showing the format of special IPv6 addresses used with the 6to4 tunnelling technique.

In Figure 1, an IPv4 domain 10 separates a first IPv6 domain 12, constituting in accordance with the present invention a first network operating in accordance with a first transmission protocol and having network addresses in accordance with a first addressing convention, from a second IPv6 domain 14, constituting in accordance with the present invention a third network also operating in accordance with the first transmission protocol and having network addresses in accordance with the first addressing convention. Hosts in the IPv4 domain 10 are IPv4 only, and hosts in the IPv6 domains 12 and 14 are IPv6 only.

The IPv4 domain 10 constitutes in accordance with the present invention a second network operating in accordance with a second transmission protocol and having network addresses in accordance with a second addressing convention. For simplifying the drawings, no IPv4 hosts are shown, and in each of the IPv6 domains 12 and 14 only one IPv6 host (28 and 30, respectively, as referred to later) is shown.

The first IPv6 domain 12 is connected to the IPv4 domain 10 via a border router 16A, and the second IPv6 domain 14 is connected to the IPv4 domain 10 via a border router 16B. The border router 16B is identical to the border router 16A, and each constitutes an interface.

The IPv4 domain 10 contains a complete domain name system (DNS) 20 including a plurality of DNS servers 22, of which only two DNS servers 22A and 22B are shown, and the IPv6 domains 12 and 14 contain respective DNS servers 24 and 26.

Suppose that a host 28 in the first IPv6 domain 12 wishes to send a packet to a host 30 in the second IPv6 domain 14. Thus, for this transaction, the host 28 is referred to as the source host 28 and the host 30 is referred to as the destination host 30 .

The source host 28 knows the name of the destination host 30, so it constructs in known manner an IPv6 DNS Request message (not shown) requesting

the IPv6 address of the destination host 30 . The source host 28 sends this DNS Request message as a recursive request to its local DNS server, which in this embodiment is the DNS server 24. The DNS server 24 will, in known manner, send a number of iterative DNS Request messages (not shown) to the DNS 20 until it learns about the DNS server 26. Finally, a DNS Request message (not shown) will go to the DNS server 26 requesting the IPv6 address of the destination host 30.

As the DNS request passes from the first IPv6 domain 12 through the border router 16A to the IPv4 domain 10, it is processed by a protocol converter (PC) 32A (see Figure 2) and undergoes IPv6/IPv4 translation. Correspondingly, as the DNS request passes from the IPv4 domain 10 through the border router 16B to the second IPv6 domain 14, it is processed by a protocol converter 32B and undergoes IPv4/IPv6 translation.

The protocol converters 32A and 32B conform to a specification known as Network Address Translation-Protocol Translation (NAT-PT). They translate between IPv4 and IPv6 addresses and keep state during the time of the session, and translation between IPv4 and IPv6 DNS Request messages and DNS Response messages, including translation of IP headers and DNS payload information is controlled by an application layer gateway (ALG). In the art, an alternative term for a DNS Response message is a DNS Answer message.

For more detailed information the reader can obtain from the Internet Engineering Task Force (IETF), the document known as draft-ietf-ngtrans-natpt-05.txt, or any variant thereof, "Network Address Translation – Protocol Translation (NATPT)", by G. Tsirtsis and P. Srishuresh.

The DNS server 26 responds to the DNS Request message for the IPv6 address of the destination host 30 with a IPv6 DNS Response message 34 (see Figure 3) having a conventional format of destination address field 36, source address field 38, and response address record 40 that contains the IPv6 address of the destination host 30.

This IPv6 DNS Response message 34 travels through the second IPv6 domain 14 to the border router 16B where it becomes converted to an IPv4 DNS Response message 42 (see Figure 4) comprising destination address field 44, source address field 46, response address record 48 and, in accordance with the present invention, additional records 50 and 52, and this message 42 travels through the

IPv4 domain 10 to the border router 16A where it becomes converted to an IPv6 DNS Response message 54 comprising destination address field 56, source address field 58 and response address record 60, and sent to the source host 28. The route that the DNS Response message (in its various forms, i.e. 34, 42 and 54) takes in the domains 10, 12 and 14 depends on the DNS configuration in each of the domains, but it will have to pass through the border router 16B and the border router 16A in that order.

For convenience, the terms "field" and "record" are used synonymously and interchangeably in this description, although in the art a field is generally taken to be a component part of a record.

When the IPv6 DNS Response message 34 is received by the border router 16B via its IPv6 network interface controller 62B, it is fed in parallel to the protocol converter 32B and to a controller 64B, and also to an encapsulator 86B and a 6to4 encapsulator 90B. The controller 64B is connected via a control line 65A to control inputs of the protocol converter 32B, the encapsulator 86B and the 6to4 encapsulator 90B and by placing a suitable address on the control line 65A selects the appropriate one of these devices.

The controller 64B (i) recognises that the received message is a DNS Response message and enables the protocol converter 32B, (ii) retrieves the IPv6 address from the response record 40 and writes this message to a storage location 66B formed by part of its internal memory 68B; (iii) writes the IPv4 address of the border router 16B, i.e. the IPv4 address of the tunnel terminating endpoint on the border router 16B, to a storage location 70B also formed by part of its internal memory 68B; (iv) receives from the protocol converter 32B the converted DNS Response message 42; appends the content of the storage location 70B as the first additional record 50, and the content of the storage location 68B as the second additional record 52; and (v) sends the resulting IPv4 DNS Response message 42 to an IPv4 network interface controller 72B for transmission over the IPv4 domain 10 to the border router 16A.

In a variant, the received IPv6 DNS Response message 34 is fed only to the controller 64B, which writes this message to a storage location 74B formed by part of its internal memory 68B. The controller 64B then (i) retrieves the IPv6 address from the response record 40 and writes this message to the storage location 66B; (ii)

writes the IPv4 address of the border router 16B to the storage location 70B; (iii) generates a modified IPv6 DNS Response message by retrieving the content of the storage location 66B and appending the content of the storage location 70B as the first additional record 50, and the content of the storage location 66B as the second additional record 52; and (iv) sends this modified IPv6 DNS Response message to the protocol converter 32B. The ALG of the protocol converter 32 processes only the header and address response record of the DNS Response message to produce the resulting IPv4 DNS Response message 42, i.e. it allows the additional records to remain unchanged.

It will be appreciated that the feeding of the received message directly to the protocol converter 32B, and the sending of an enabling signal on the control line 65A to the protocol converter 32B by the controller 64B is logically equivalent to the sending of the received message to the protocol converter 32B by the controller 64B in the above variant, and constitutes sending the message to the protocol converter 32B in accordance with the present invention.

When the IPv4 DNS Response message 42 is received by the border router 16A via its IPv4 network interface controller 72A, it is fed in parallel to the protocol converter 32A and to a controller 64A.

The controller 64A (i) receives from the protocol converter 32A the output IPv6 DNS Response message comprising destination address field 56, source address field 58, response address record 60, and additional records 50, 52; and (ii) retrieves the IPv6 address from the second additional record 52, i.e. the true IPv6 address of the destination host 30, and inserts it into the response address record 60 of that output message instead of the IPv4-compatible IPv6 address for the destination host 30, which the protocol converter 32A had generated. The controller 64A then strips off the additional records 50, 52, and sends the resulting IPv6 DNS Response message 54 (see Figure 5) to an IPv6 network interface controller 62A of the border router 16A for transmission to the source host 28.

Additionally, the controller 64A is arranged to retrieve the IPv4 address of the tunnel terminating endpoint from the first additional record 50, to create a mapping of the IPv6 address of the destination host 30 to the address of the IPv4 tunnel terminating endpoint, and to store this mapping, i.e. create an entry, in an

IPv6/tunnel endpoint table 76A formed by part of an internal memory 68A of the controller 64A and constituting a look-up table of the present invention.

In a variant, the additional records 50, 52 are stripped from the DNS Response message prior to the retrieval of their contents. In another variant, the additional records remain attached to the DNS Response message, but this is not as efficient as stripping the additional records. In this present embodiment, each entry in the IPv6/tunnel endpoint table 76A comprises a first element 78A comprising the IPv6 address of a corresponding destination host 30, a second element 80A comprising the IPv4 address of the tunnel terminating endpoint, i.e. of the border router 16B, and third and fourth elements 82A and 84A, to be described later.

Upon receipt of the resultant IPv6 DNS Response message 54, the source host 28 retrieves the IPv6 address from its address record 60 and stores it for use in sending data packets to the destination host 30.

In known manner, the source host 28 generates for each of these data packets a header including a source address field and a destination address field, and writes the retrieved IPv6 address into the destination address field.

Upon receipt of each of these data packets at the border router 16A, the controller 64A retrieves the destination address, and, in accordance with the retrieved destination address, accesses the IPv6/tunnel endpoint table 76A. If there is a match with the contents of a first element 78A of an entry, the controller 64A retrieves the corresponding IPv4 tunnel terminating endpoint from the second element 80A of that entry, and commands an encapsulator 86A to encapsulate the packet in an IPv4 packet. Thus, the encapsulator 86A appends an IPv4 header into which it has inserted its own IPv4 address into the source field, and the retrieved IPv4 tunnel terminating endpoint address into the destination field. In this embodiment the encapsulator 86A stores its own IPv4 address for this use. In variants, the controller 64A stores this IPv4 address, and passes it, together with the retrieved IPv4 tunnel terminating endpoint address, to the encapsulator 86A when commanding it to perform encapsulation.

In this embodiment, the encapsulator 86A is arranged to receive the packet directly from the IPv6 network interface controller 62A of the border router 16A, but it does not perform encapsulation until commanded by the controller 64A. In a variant, the controller 64A receives the packet directly from the IPv6 network

interface controller 62A and passes it to the encapsulator 86A if there is a match. In practice, when the border router 16A receives a packet the controller 64A writes it into a storage location of its internal memory 68A, and the controller 64A will pass to the encapsulator 86A the address of the relevant storage location, together with an instruction for the encapsulator 86A to access that storage location.

Upon receipt of the encapsulating IPv4 packet at the border router 16B, an un-encapsulator 88B of the border router 16B strips off the IPv4 header and retrieves the payload of that IPv4 packet, i.e. un-encapsulates the original IPv6 packet from the source host 28, and sends that IPv6 packet to the destination host 30. The controller 64B also creates a mapping (in its IPv6/tunnel endpoint table 76B) of the IPv6 address of the source host 28 and the tunnel originating endpoint, i.e. the IPv4 address of the originating border router 16A, these being respectively retrieved from the source address field of the IPv6 header and the source address field of the IPv4 header.

When the destination host 30 returns a Reply packet, a controller 64B of the border router 16B retrieves the destination address, "IPv6 host 28", from the received Reply packet, accesses its IPv6/tunnel endpoint table 76B in accordance with the retrieved destination address (i.e. seeking a matching element 78B), retrieves the corresponding IPv4 address (element 80B), and commands an encapsulator 86B to encapsulate the Reply packet in an IPv4 packet addressed to an un-encapsulator 88A of border router 16A using the IPv4 tunnel originating endpoint address just retrieved from the element 80B of the IPv6/tunnel endpoint table 76B. Upon receipt of this IPv4 packet at the border router 16A, the un-encapsulator 88A performs un-encapsulation to retrieve the Reply packet, and the border router 16A then sends the retrieved Reply packet to the source host 28.

The source host 28 and the destination host 30 are now in a session in which IPv6 packets pass between them via the tunnel just established between the border routers 16A and 16B.

The above described mechanism provides for an IPv6 host, which is in an isolated IPv6 domain, to communicate with another IPv6 host, which is in another isolated IPv6 domain, via an intermediate IPv4 domain, without any knowledge of where that other IPv6 host is, and without the source IPv6 host needing to do anything different from a standard communication procedure with another IPv6 host

within its own IPv6 domain. The DNS server local to the source IPv6 host makes a Request via the IPv4 domain to the IPv6 DNS server that is on the same network as the destination IPv6 host, and the border routers automatically set up respective mappings associating the tunnel endpoint and the IPv6 address of the IPv6 hosts behind the border routers.

In alternative embodiments, some of the entries in the IPv6/tunnel endpoint table 76A can be created by the administrative personnel of the network operator. This is known as manual configuration of a tunnel, and the tunnel is permanent until changed at a later date by the administrative personnel.

As shown in Figure 2, the border router 16A also comprises a 6to4 tunnelling encapsulator 90A (and 6to4 tunnelling un-encapsulator 92A) and can thus interwork with a border router which is similarly enabled, although in variants these may be omitted. The special IPv6 addresses 94 (see Figure 6) used for this technique have a three-part format of which a first part 96 having thirty two bits is a prefix uniquely identifying that the packet is to be tunnelled by the 6to4 tunnelling technique, a second part 98 having thirty two bits is the IPv4 address of the 6to4 tunnel endpoint, and the third part 100 having sixty four bits is known as the interface ID which is the modified MAC address of the destination host. The second part 98 constitutes a predetermined subaddress field of the destination address of the present invention.

In variants having a different tunnelling encapsulator, a different respective prefix is used for the same purpose.

In some variants of these embodiments, the controller 64A is arranged to recognise the presence of this prefix within the retrieved destination address of a received packet, to retrieve from its second part 98 the IPv4 address of the 6to4 tunnel endpoint and to command the 6to4 tunnelling encapsulator 90A to handle the received packet using the retrieved IPv4 address. This constitutes deriving the second type address directly. Alternatively, the 6to4 tunnelling encapsulator 90A is arranged to retrieve the special IPv6 address and to extract from its second part 98 the IPv4 address of the 6to4 tunnel endpoint.

Where, as mentioned above, the controller 64A is arranged to perform prefix recognition, the prefix to be recognised is stored in a storage location of its internal

memory 68A, and this storage location can be an entry or part of an entry of the IPv6/tunnel endpoint table 76A.

The un-encapsulators 88B and 92B have respective IPv4 addresses, which are used by the corresponding encapsulators 86A and 90A in generating their respective encapsulated packets.

In the preferred arrangement of border router having a plurality of different encapsulators, e.g. 86, 90, the controller 64A accesses the IPv6/tunnel endpoint table 76A in accordance with a set of match criteria to cover the range of possible situations. These are

(a) a tunnel has already been established by the above described DNS Request technique and a IPv6 destination-specific IPv6/IPv4 entry exists in the IPv6/tunnel endpoint table 76A;

(b) a tunnel has already been established by one of the known tunnelling techniques and an IPv6/IPv4 entry exists in the IPv6/tunnel endpoint table 76A, of which the first element 78A has a first part in the form of a specific prefix corresponding to that tunnelling technique;

(c) network management personnel have manually configured the border router 16 to define a tunnel to a specific IPv6 destination host using 6to4 (or 6over4) to another border router (which might be the border router 16B or a different border router (not shown) associated with a further IPv6 domain (not shown)), and for this case the IPv6/tunnel endpoint table 76A has an entry whose first element 78A is the IPv4-compatible (or IPv4-mapped) address of that destination host;

(d) network management personnel have manually configured the border router 16A to define a tunnel to unspecified IPv6 destination hosts using 6to4 (or 6over4) to another border router (which might be the border router 16B or a different border router associated with a further IPv6 domain), and for this case the IPv6/tunnel endpoint table 76A has an entry having its first element 78A in the form of the 6to4 (or 6over4) prefix followed by the IPv4 address of that other border router followed by null characters, and in some variants the second element 80A of this entry contains null characters, whilst in yet other variants the second element 80A of this entry contains the IPv4 address of that other border router; and

(e) the table contains an entry whose first element 78A is a generic IPv4-compatible or IPv4-mapped IPv6 address, i.e. its first eighty bits are all zeros and the

final thirty two (or in a variant, forty eighty) bits are null characters (zeros), the second and fourth elements 80A and 84A contain null characters, and a third element 82A contains the identifier "PC".

The controller 64A uses its IPv6/tunnel endpoint table 76A to determine the appropriate handling of a received packet in the following manner.

If the controller 64A finds an entry having a first element 78A matching the complete retrieved destination address, then the content of the second element 80A of that entry is retrieved and used as the IPv4 destination address, i.e. that of the border router 16B, the tunnel endpoint. Additionally, the content of the third element 82A of that entry is retrieved and used as a check that the retrieved IPv4 destination address and the packet received by the border router 16A are to be processed by the encapsulator 86A. The content of the third element 82A is either an identifier for an encapsulator 86A, 90A (e.g. "EN") or an identifier for the protocol converter 32A (e.g. "PC"). As a further check, the fourth element 82A of that entry contains an identifier for the encapsulation type. In other words, for an entry matching the complete retrieved destination address, as just described, the encapsulation type identifier will be "DNS" to signify that the encapsulator 86A is to be used.

If the controller 64A finds an entry whose first element 78A has the first thirty two bits matching the first thirty two bits, i.e. the special 6to4 prefix part, of the retrieved destination address, then the third and fourth elements 82A and 84A of that entry are checked ("EN" and "6to4", respectively), the IPv4 destination address is retrieved from the second element 80A of that entry and sent with the packet received by the border router 16A to be processed by the encapsulator 90A. This constitutes indirectly retrieving the second type address from a predetermined subaddress field of the destination address.

If the retrieved destination address is either IPv4-compatible or IPv4-mapped, i.e. the packet is to be protocol converted for an IPv4 destination, then its first eighty bits will be all zeros, and the following sixteen bits will be either all zeros if the address is IPv4-compatible, or all ones if the address is IPv4-mapped. Thus the controller 64A checks to see whether its IPv6/tunnel endpoint table 76A contains an entry whose first element 78A has its first eighty bits all zeros The content of the second element 80A of such an entry will be all null characters, because no tunnel is involved, and the content of the fourth element 84A of such an entry will be all null

characters, because no encapsulation is involved. The content of the third element 82A of that entry is retrieved and used as a check that the retrieved IPv4 destination address and the packet received by the border router 16A are to be processed by the protocol converter 32A. In this case, the content of the third element 82A is an identifier for the protocol converter 32A (e.g. "PC").

In other variants, the controller 64A is arranged to access the IPv6/tunnel endpoint table 76A, as before, and only command the 6to4 tunnelling encapsulator 90A upon detecting a match with an entry, and in this case the controller 64A passes the special IPv6 address to the 6to4 tunnelling encapsulator 90A, or alternatively the controller 64A extracts the IPv4 address of the 6to4 tunnel endpoint and passes it to the 6to4 tunnelling encapsulator 82A, or yet again the controller 64A, upon detecting such a match, retrieves the element 80A of the matching entry. This element 80A contains the IPv4 address of the 6to4 tunnel endpoint, which was inserted into that element 80A by the controller (or manually) upon creation of that entry.

In the above embodiment, the local DNS server for the source host 28 is the IPv6 DNS server 24, but in alternative embodiments it can be one of the IPv4 servers 22 of the DNS 20. In such alternative embodiments, although the host 28 can send a DNS Request message for obtaining the IPv6 address of the host 30 and, by the present invention, establish a tunnel across the IPv4 domain10, the situation is not symmetrical in that the host 30 cannot act as a source and establish a corresponding tunnel across the IPv4 domain10 to the host 28. For an IPv6 host to be contactable, i.e. act as destination, by the method of the present invention, it is necessary for the DNS server local to that IPv6 host to be an IPv6 DNS server on the same IPv6 domain as that IPv6 host, because the DNS Response message has to pass through the border router adjacent to that IPv6 host in order that the tunnel can be established. In other words, the DNS Request message has to pass through the IPv4 domain and not stop at an IPv4 DNS server acting as the local DNS server for the intended destination IPv6 host.

CLAIMS

1. An interface for use between a first network operating in accordance with a first transmission protocol and having network addresses in accordance with a first addressing convention, herein referred to as first type addresses, and a second network operating in accordance with a second transmission protocol and having network addresses in accordance with a second addressing convention, herein referred to as second type addresses, the interface having both a first type address and a second type address and comprising:

a protocol converter arranged to convert a message having a format in accordance with the first transmission protocol, herein referred to as a first type message, into a message having a format in accordance with the second transmission protocol, herein referred to as a second type message, and vice versa;

means for encapsulating arranged to respond to receipt of a second type address together with a first type message by encapsulating that received first type message as the payload of a resulting encapsulating second type message, using that received second type address as the destination address of the resulting encapsulating second type message and using the second type address of the interface as the source address of the resulting encapsulating second type message;

means for un-encapsulating a second type message to retrieve its payload; and

an interface controller arranged to respond to receipt by the interface of a first type message from the first network by

examining the destination address of that first type message received from the first network,

sending to the protocol converter that first type message received from the first network if its destination address is of a first predetermined format,

else, deriving, directly or indirectly, from the destination address of that first type message received from the first network, a second type address for use by the encapsulating means as the destination address of a resulting encapsulating second type message, and

sending to the encapsulating means the derived second type address together with that first type message received from the first network.

2. An interface as claimed in claim 1, wherein the controller is arranged to derive the second type address directly by retrieving it from a predetermined subaddress field of the destination address.

3. An interface as claimed in claim 1, wherein the controller is arranged to derive the second type address indirectly by accessing, in accordance with the destination address, a look-up table having entries in the form of a first type address associated with a second type address, and retrieving the second type address of an entry having a first type address matching the destination address.

4. An interface as claimed in claim 3, wherein each entry of the look-up table comprises a field for containing an identifier for identifying whether the controller is to send that first type message received from the first network to the protocol converter or to the encapsulating means.

5. An interface as claimed in any one of claims 1 to 4, wherein the encapsulating means comprises a plurality of different encapsulators, each encapsulator being arranged to operate in accordance with a respective encapsulation type, and the controller is arranged to send that first type message received from the first network to the appropriate one of said plurality of different encapsulators in dependence upon the format of the destination address of that first type message.

6. An interface as claimed in claim 5, when dependent on either claim 3 or claim 4, wherein each entry of the look-up table comprises a field for containing an identifier for identifying a type of encapsulation.

7. A method of operating an interface between a first network operating in accordance with a first transmission protocol and having network addresses in accordance with a first addressing convention, herein referred to as first type addresses, and a second network operating in accordance with a second transmission protocol and having network addresses in accordance with a second

addressing convention, herein referred to as second type addresses, the method comprising the steps of:

examining the destination address of a first type message received from the first network; and

if the destination address of that received first type message is of a first predetermined format, protocol converting that received first type message;

else, encapsulating that received first type message in accordance with the second transmission protocol, using, as the destination address of a resulting encapsulating second type message, a second type address derived, directly or indirectly, from the destination address of that received first type message.

8. A method of operating an interface between a first network operating in accordance with a first transmission protocol and having network addresses in accordance with a first addressing convention, herein referred to as first type addresses, and a second network operating in accordance with a second transmission protocol and having network addresses in accordance with a second addressing convention, herein referred to as second type addresses, the method comprising the steps of:

examining the destination address of a first type message received from the first network; and

if the destination address of that received first type message is of a first predetermined format, protocol converting that received first type message; and

if the destination address of that received first type message is of a second predetermined format, encapsulating that received first type message in accordance with the second transmission protocol, using, as the destination address of a resulting encapsulating second type message, a second type address derived, directly or indirectly, from the destination address of that received first type message.

9. A method as claimed in claim 8, wherein the second predetermined address format includes an identifier identifying an encapsulation type.

10. A method as claimed in any one of claims 7 to 9, wherein the first predetermined format includes a first predetermined portion whose content identifies that received first type message as suitable for protocol conversion.

11. A method as claimed in claim 10, wherein the first predetermined format also includes a second predetermined portion whose content constitutes the second type address used as the destination address of a resulting encapsulating second type message.

12. A method as claimed in any one of claims 7 to 11, wherein the second type address is derived directly by retrieving it from a predetermined subaddress field of the destination address.

13. A method as claimed in any one of claims 7 to 12, wherein the examining step comprises the substeps of retrieving the destination address from the received first type message, and accessing a look-up table in accordance with the retrieved destination address.

14. A method as claimed in claim 13, for use when the look-up table comprises entries in the form of a first type address associated with a second type address, and wherein retrieval of the second type address of an entry having its first type address matching the destination address constitutes indirectly deriving the second type address from the destination address of that received first type message.

15. A method as claimed in claim 14, for use when the look-up table entries include a first identifier field containing an identifier identifying whether that first type message is to be protocol converted or encapsulated, and including the steps of retrieving the identifier from the first identifier field of the entry having its first type address matching the destination address, and checking that the retrieved identifier is consistent with whichever of protocol conversion or encapsulation is to be performed upon that received first type message.

16. A method as claimed in either claim 14 or claim 15, for use when the look-up table entries include a second identifier field containing an identifier identifying an encapsulation type, and when there is a plurality of encapsulation types available, and including the steps of retrieving the identifier from the second identifier field of the entry having its first type address matching the destination address, and checking that the retrieved identifier is consistent with the type of encapsulation to be performed upon that received first type message.

17. An interface for use between a first network operating in accordance with a first transmission protocol and having network addresses in accordance with a first addressing convention and a second network operating in accordance with a second transmission protocol and having network addresses in accordance with a second addressing convention, the interface being substantially as herein described with reference to the drawings.

18. A method of operating an interface between a first network operating in accordance with a first transmission protocol and having network addresses in accordance with a first addressing convention and a second network operating in accordance with a second transmission protocol and having network addresses in accordance with a second addressing convention, the method being substantially as herein described with reference to the drawings.

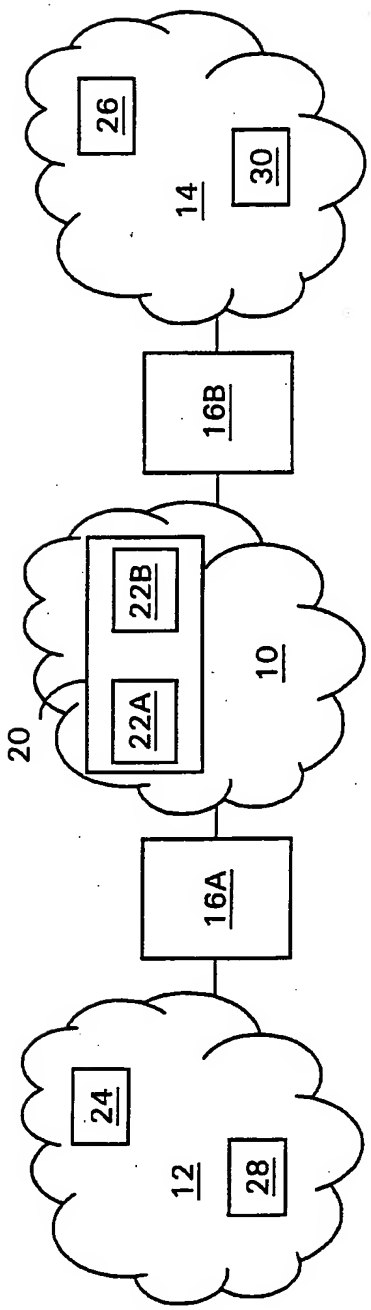


Fig.1

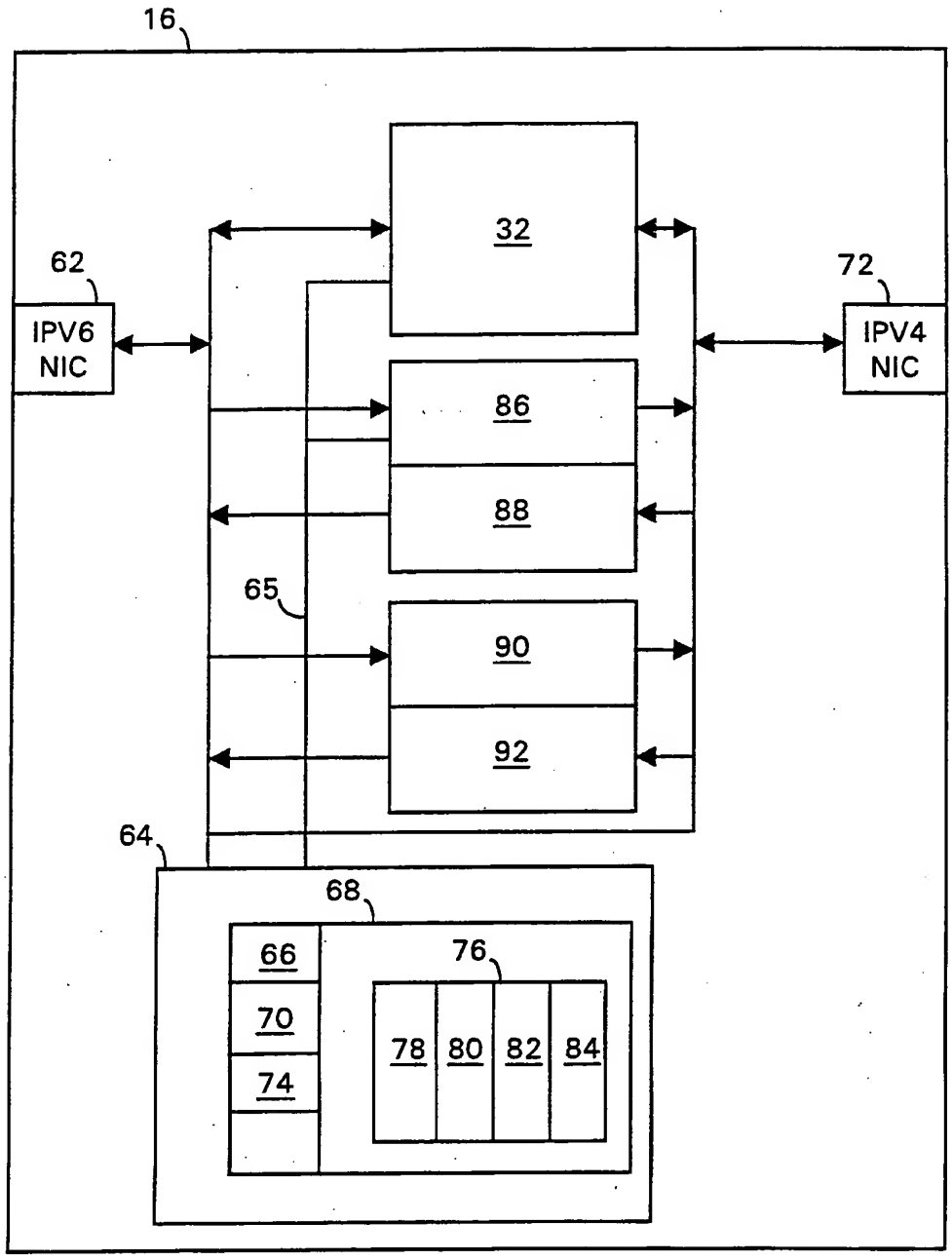

16
32
62
IPV6 NIC
72
IPV4 NIC
86
88
65
90
92
64
68
66
76
70
74
78
80
82
84

Fig. 2

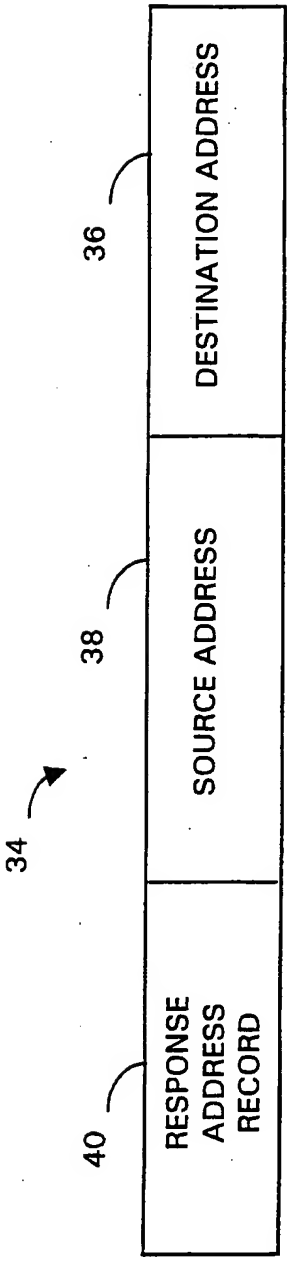

34
36
DESTINATION ADDRESS
38
SOURCE ADDRESS
40
RESPONSE ADDRESS RECORD

Fig. 3

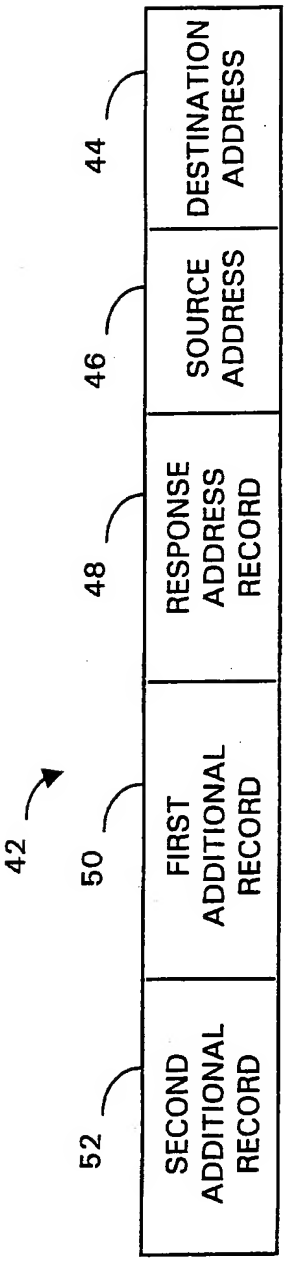

42
44
DESTINATION ADDRESS
46
SOURCE ADDRESS
48
RESPONSE ADDRESS RECORD
50
FIRST ADDITIONAL RECORD
52
SECOND ADDITIONAL RECORD

Fig. 4

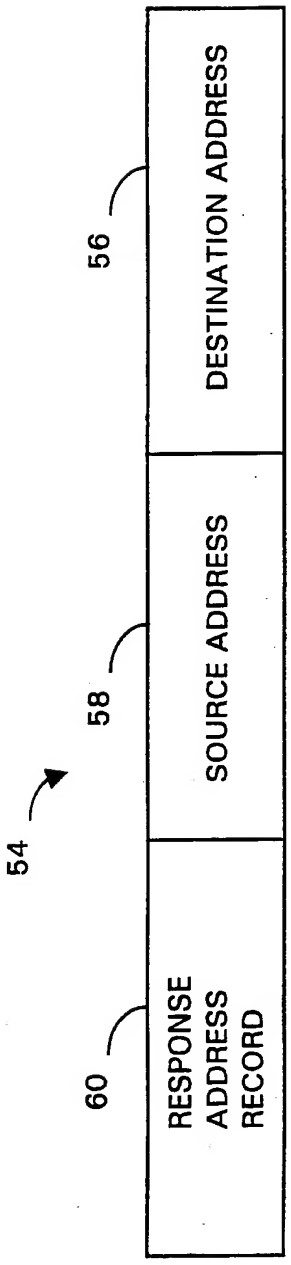

54
56
58
60
DESTINATION ADDRESS
SOURCE ADDRESS
RESPONSE ADDRESS RECORD

Fig. 5

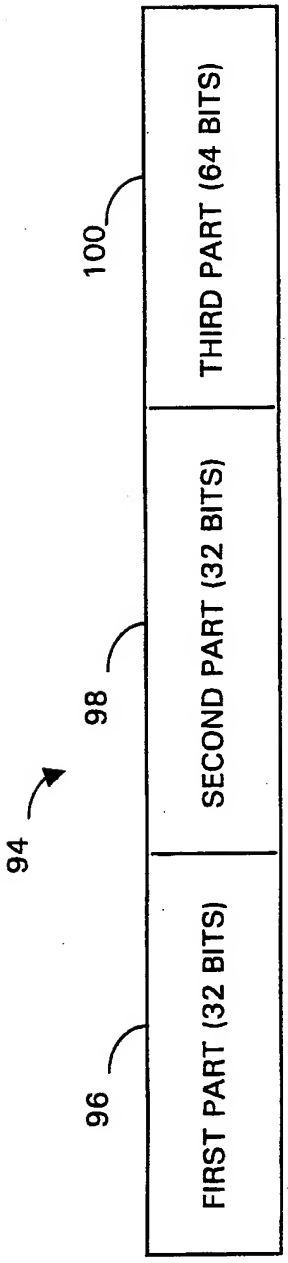

94
96
FIRST PART (32 BITS)
98
SECOND PART (32 BITS)
100
THIRD PART (64 BITS)

Fig. 6